The MMSFORTH Newsletter

MILLER MICROCOMPUTER SERVICES
61 LAKE SHORE ROAD, NATICK, MASS. 01760
(617) 653-6136

July-August 1980

Vol.1 No.3

BYTE PRINTS SPECIAL FORTH ISSUE!

Hurry up and get your copy while tbey last! The August issue of BYTE Magazine is a collector's item for Forth fans everywhere! Several good articles, tables and an excellent editorial by Gregg Williams provide information on Forth from introductory examples through <BUILDS and DOES> . It includes our full presentation of BREAKFORTH, complete with source code for users of Cassette MMSFORTH. (Byte has had so many Forth articles submitted that it will run some in following issues.) As a convenience, MMS will send a copy of the August BYTE with your next order; add $4.00, including the extra postage. Your regular BYTE dealer can supply it for a mere $2.50, so we recommend that alternative if it is available.

In addition to the many good articles in BYTE, we also liked a recent article by Michel Mannoni of Forth, Inc. which appears on Page 175 of the July 19th issue of Electronic Design magazine. We found a few small errors - can you?

MEET THE CREW AT MMS! (Editorial)

Thanks for the vote of confidence, often of eagerness if not outright addiction, for our MMSFORTH Newsletter! By sending in your $10.00 check you have told us we are doing it at least reasonably right for a lot of users. Given your encouragement, we intend to keep it coming.

Speaking of subscriptions, we strongly recommended FIG, the Forth Interest Group, in Newsletter #1. FIG's membersbip cost has increased, and it's still worth it. The new price is $12.00 for this year's dues (including six issues of FORTH DIMENSIONS); an additional $6.00 gets you the first six issues, as well. For overseas air, make those prices $15.00 and $8.00, respectively.

Some of you who have chatted with and met some of the MMSFORTH crew have suggested that we share the secret with all our active users. We are your usual crew of TRS-80-happy programmers, mixing business and pleasure by forsaking most pleasures other than business! MMS believes that microcomputers and powerful programming tools such as MMSFORTH can provide affordable computing power to individuals and small businesses while still assisting in the big shops. True to our beliefs, we bought our first TRS-80 the day after they were announced in August 1977, and we use them and a mess of peripherals to build a state-of-the-art cottage industry.

From 9 A.M. to 9 P.M. Jill and Dick (known as A. Richard to his friends) Miller are apt to be found at the MMS office, which they intend to keep in their cottage/home on the shore of Lake Cochituate near Boston. They are assisted by a friendly blue parakeet named Alistair Cook (known as Cookie to his friends) who flies about tbe house saying "I want to play with the computer", "Garbage in, garbage out", and dropping other useful hints. In prior incarnations Dick has been an electrooptical physicist sending sophisticated garbage to the Moon, and an environmental activist getting it out of his favorite lake. Now Dick writes this Newsletter and the MMSFORTH literature, handles most marketing activities and company communications, and is getting better and better at using and explaining Forth. Once a month Dick forgets his job and relaxes - as President of TRUGEM, the very first TRS-80 Users Group.

Jill is Dick's pretty wife, which makes him her ugly husband. She has some 15 years of systems analysis and business programming experience in many languages, mostly on IBM mainframe systems. She marvels that so many of her co-programmers haven't yet noticed the professional capabilities of the microcomputer. Personally she finds it a much more satisfying working environment - now that she is equipped with Forth! Once a week Jill takes a few hours off to cook great food, tend to her beehives and garden (she has a green thumb and a degree in botany), or to participate in her quilting group. This time of the year Dick and Jill often obey Cookie, who also says, "Go jump in the lake!".

The hero of this intimate expose is Tom Dowling, Forth programmer extraordinary. Tom is the author of MMSFORTH and toils over it to keep Dick, Jill and you working in the perfect language - if he has his way. Tom has created five versions of Forth before ours; his knowledge of Forth and the Millers' knowledge of the TRS-80 have resulted in the heady blend we all use today. Unlike most geniuses, Tom doesn't live in an ivory tower. He lives in a granite tower, instead - and we mean literally! Once a year Tom, his wife Sunny, Jill and Dick all enjoy sailing on 100-year-old windjammer schooners off the Maine coast, as a change from jumping in the lake or slaving over their hot keyboards.

Others - some of you - also are working on upcoming MMSFORTH products. In future issues we will introduce you to more members of the MMS team, our network of talented users in this growing cottage industry.

-- Dick Miller, Editor 4th Class

SETTING FORTH (for beginners)

KALEIDOSCOPE:

You'll like this one! Following a recent MMS presentation on Forth, Dave Huntress challenged us with his roughed-out BASIC code for this routine. "Tbis program draws a neat display," he said, "but in BASIC it runs too slow to be interesting. Can Forth do graphics?"

Can it ever! Our previous SETTING FORTH programs used graphics, but not the TRS-80 Level II-style graphics set which also is supported in MMSFORTH (in single- and double-width element sizes!). Here we use MMSFORTH's single-element word, ESET. Dave decided to go Forth as soon as he saw the dramatic speed-up. Others may be even more impressed with the structured readability of the Forth code - once you understand Forth, that is! Now we've got a great introduction for our next talk, and so do you! Because it's a one-block program it is particulary easy to demonstrate a point and then return to execution. After 81 EDS and 81 LOAD, just Break during the drawing phase and enter V to re-View the screen; enter KSCOPE to restart your kaleidoscope!

Now that you are getting better at writing in Fortb, we won't explain this one in detail. Instead, we spent extra effort arranging the source block neatly and picking meaningful word names to aid in your reading of our code. Here are two clues to help. DX and DY get reset for each new set of 4 lines, and can only have values of 1, -1, or 0. We use V1.9's new OR command to decide if any of the four proposed lines will exceed the screen borders, then reverse the trutb with a NOT to see if it fits, not if it overlaps. (You can insert OR in Block 18 Lines 1 and 2 of your pre-V1.9 MMSFORTH System and reprecompile basic MMSFORTH, as we did, or you can insert it near the top of the KALEIDOSCOPE block:

```
CODE OR   HL POP  DE POP  L  A  MOV  E ORA  A L MOV  H A MOV  D ORA
A H MOV  PSH  )
```

Take it from there!

BLOCK : 81

```
 0 ( KALEIDOSCOPE PROGRAM, FROM BASIC PROGRAM BY D. HUNTRESS 7/80 )
 1 : TASK ;      32 LOAD ( RANDOM NUMBERS )    33 LOAD ( GRAPHICS )
 2 64 CONSTANT X   23 CONSTANT Y    ( OFFSETS TO CENTER OF SCREEN )
 3 0 VARIABLE LENGTH   0 VARIABLE DX   0 VARIABLE DY ( INCREMENTS )
 4
 5 : ?FIT   X DX @ +  0 <    Y DY @ +  0 <
 6          X DX @ + 64 >    Y DY @ + 23 >   OR OR OR  NOT ;
 7 : 4-LINES   LENGTH @ 0 DO  ?FIT   IF DX @  ' X +!  DY @  ' Y +!
 8   Y X ESET  Y 127 X - ESET  47 Y - X ESET  47 Y - 127 X - ESET
 9   ELSE LEAVE   THEN LOOP ;
10
11 : PAUSE   20000 0 DO LOOP ;              ( PAUSE FOR 2 SECONDS )
12 : MESSAGE 14 2 DO  I DUP 2 * 13 + PTC " MMSFORTH"  LOOP  PAUSE ;
13
14 : KSCOPE   BEGIN  CLS  70 0 DO  3 RND 2 - DX !  3 RND 2 - DY !
15   30 RND LENGTH !  4-LINES  LOOP  MESSAGE  0 END ;      KSCOPE
```

GET-TOGETHER

Consider sharing your questions and answers with a MMSFORTH User Group, or contact MMS for help in starting one in your metropolitan area. Here is our present list of contacts for local MMSFORTH User Groups:

Morris Herman, 503 Rosario Drive, Santa Barbara CA 93110 (805/964-7144).
Rich Royea, 6456 Lubau, Woodland Hills CA 91367 (213/704-6859).
Ed Laughery, 1222 Jason Drive, Denham Springs LA 70726 (504/665-7537).
Jim Gerow, 1630 Worcester Road, Framingham MA 01701 (617/443-9521 x3562 days, 617/872-1882 eves.).
Kim Watt, Box 1013, Berkeley MI 48072 (313/288-9422).
Bob Zwemer, 6408 South Washington, Lansing MI 48910 (517/393-9287).
Larry Goforth, 10203-J Golden Meadow, Austin TX 78758 (512/836-0981).
Jim Shepard, 16210 Arbor Downs Drive, Dallas TX 75248 (214/661-9702).
Paul van der Eijk, 4910 Fran Place #204, Alexandria VA 22312 (703/354-7443).
Rod Proctor, 13520 N.E. 29th Place, Bellevue WA 98005 (206/885-4171 days, 206/883-1923 eves.). Rod also is on THE SOURCE.

NOTE: Program trading is one popular facet of these meetings, but NOT commercial programs and WITHOUT MMSFORTH SYSTEMS aboard! Promote legitimate sharing, discourage pirating, and take care not to jeopardize your own MMSFORTH serial number.

ABSOLUTE ASSEMBLER (no extra charge!):
by Jill Miller, programming by Tom Dowling

Foreword:

A significant number of MMSFORTH users have requested a cross-compiler; that is, a method to write in MMSFORTH but to compile into code which then can be run on a TRS-80 without a MMSFORTH system. MMSFORTH is not set up for this, and there are both marketing and technical problems in doing this well on a general basis. We don't want to run a weak version of MMSFORTH in DOS or CP/M. MMS provides custom cross-compiling services for this and other purposes, and finds that each job requires a somewhat different approach for optimal results. And we are understandably cautious about providing a few MMSFORTH users with an inexpensive tool to keep their customers away from exposure to Forth instead of bringing them systems which utilize the full power of its language/operating system.

While awaiting a perfect solution to this dilemma, here is a compromise tool at a price that's right: our new Absolute Assembler written in MMSFORTH which can be used by you to create SYSTEM cassettes for running without MMSFORTH - or for loading under conventional disk operating systems with TAPEDISK (in TRSDOS), LMOFFSET (in Apparat's NEWDOS+) or a similar routine. MMS used it to create the additional command modules in the expanded version of NEWDOS+ for the CORVUS 10-megabyte hard disk drive. It will be the most versatile assembler you own; it embodies a number of Forth constructs and in experienced hands it can be used to redefine your favorite MMSFORTH words - or anything else!

- Editor

Herewith is presented an Absolute Assembler and a routine to be used to create SYSTEM tapes. Together these two MMSFORTH routine give the user the ability to create TRS-80 load modules, which can be used without Forth under TRSDOS or Level II BASIC.

The Absolute Assembler consists of four blocks. The first block defines the special variables for setting up the Absolute Assember. The second and third blocks define the 8080 Absolute Assembler. (The Z-80 Assembler instructions could be substituted by those who have the Floating-Point/Z-80 Assembler Package.) The fourth block sets up the logic constructs available to the programmer and loads the two blocks of the "TBUG Dump" which allows system tapes to be created.

All functions used in the Absolute Assembler must be defined using the Absolute Assembler; i.e., standard FORTH vocabulary words are NOT available. The advantage of the MMSFORTH Absolute Assembler over other available assemblers is that it allows the user to debug an assembly language program incrementally, one routine at a time, under MMSFORTH using CODE vocabulary words, then with a few minor changes to create a load module, and it provides most of the Forth logic constructs. Since the source for the Absolute Assembler is provided, further enhancements such as macro ability can be added.

Functions supplied with the Absolute Assembler:
1) Full 8080 Assember. (If you own the Floating Point/Z-80 Assembler Package, you may wish to modify it using the example of the 8080 Assembler to get a Z-80 Absolute Assembler.)
2) Subroutines can be created and called by name.
3) Routines and memory locations can be labeled.
4) The following MMSFORTH logic constructs are supplied:
   a) IF...ELSE...THEN
   b) BEGIN...END
   c) WHILE...PERFORM...PEND
   Note: Because DO...LOOP uses Forth's Return Stack, it cannot easily be implemented in the Absolute Assembler. However, loading and decrementing a register (see example) can serve as well.
5) Program can be located anywhere in RAM.
6) Name the program.
7) Specify any desired entry point.

The Absolute Assembler work area is originally set at 2K (2048) bytes, but can be easily adjusted to any size permitted by available memory in your TRS-80. The Assembler plus the dump routines take 4045 bytes.

To Use the Absolute Assembler:
1) Create and debug your program with regular MMSFORTH CODE words (no FORTH vocabulary words are allowed).
2) At the beginning of the program insert ABS-ASMB to invoke the Absolute Assembler vocabulary. Specify a starting location for the completed program in memory by storing its address in ORG . Call Assembler initialization, INITASMB .
3) Change CODE and ; definitions to subroutines as necessary. Change : definition of last word (program name) to LABEL .
4) Add JMP as last instruction in program.
5) To create a SYSTEM tape:
   entry-point TDUMP name

Important variable names in the Absolute Assembler:
1) ORG is the origin address of the program. It must be set at the beginning of the program.
2) PC is the Position Counter which points to the next available memory location. Usually it is incremented by the Absolute Assembler, but the programmer may wish to manipulate it himself (for example, to set aside a large data area).
3) BAREA is the start location of memory set aside for storing the binary program. You will want to enlarge the number following it for any program which is larger than 2K (2048 bytes).

Line-by-line (almost) examination of the Absolute Assembler:
First Block (12): This block is in DECIMAL.
Lines 2 & 3 set up variables for the Absolute Assembler.
ORG indicates the memory location from which the program will execute.
BAREA is the work area where the binary program will be built. Its original size is 2048 bytes (2046, + 2 bytes from the VARIABLE definition itself).
PC is the Program Counter or Position Counter. It keeps track of how many bytes from the beginning of your program you have filled with program.
OFFSET holds the difference between the contents of ORG and the address of BAREA. It allows the Absolute Assembler to calculate correct JUMP addresses.
Line 4 sets up the special vocabulary for the Absolute Assembler.
Lines 5-8 define the words , and C, for the Absolute Assembler. They will do the same thing as the FORTH words of the same name, but in the work area for the Assembler.
Line 9 defines a word to put a label on the next location. To label a call location in ROM:
(address) CONSTANT KOUT

BLOCK : 12

```
 0 ( ABSOLUTE ASSEMBLER ROUTINES 1 OF 4 )     : TASK ;
 1 ( COPYRIGHT 1980, MMS/T.DOWLING;  FOR MMSFORTH LICENSEES ONLY! )
 2 0 VARIABLE ORG  0 VARIABLE BAREA 2046 H +!
 3 0 VARIABLE PC   0 CONSTANT OFFSET
 4 VOCABULARY ABS-ASMB          ABS-ASMB DEFINITIONS
 5 CODE C, PC LHLD HL INX PC SHLD XCHG ' OFFSET LHLD DE DAD
 6   DE POP HL DCX E M MOV NEXT
 7 CODE ,  PC LHLD HL INX HL INX PC SHLD XCHG ' OFFSET LHLD
 8   DE DAD DE POP HL DCX D M MOV HL DCX E M MOV NEXT
 9 : LABEL PC @ CONSTANT ;
10 : BTABLE LABEL BEGIN BL WORD HERE @ 8705 = IF 1 ELSE
11   HERE NUMBER DROP C, 0 THEN END ;
12 : INITASMB ORG @ PC ! BAREA ORG @ - (') OFFSET !
13   BAREA 2048 ERASE ;
14 : $ 37 WORD HERE 1+ PC @ OFFSET + HERE C@ DUP PC +! MOVE ;
15 BASE C@  HEX
```

BLOCK : 13

```
 0 ( ABSOLUTE ASSEMBLER 2 OF 4 )
 1 : 1BY C;: C, ;
 2 2F 1BY CMA 3F 1BY CMC 27 1BY DAA F3 1BY DI FB 1BY EI
 3 76 1BY HLT 00 1BY NOP E9 1BY PCHL 17 1BY RAL 1F 1BY RAR
 4 C9 1BY RET 07 1BY RLC 0F 1BY RRC F9 1BY SPHL 37 1BY STC
 5 EB 1BY XCHG E3 1BY XTHL
 6 : 2BY C;: C, C, ;    : BCON CCONSTANT ;
 7 CE 2BY ACI C6 2BY ADI E6 2BY ANI FE 2BY CPI DB 2BY IN
 8 F6 2BY ORI D3 2BY OUT DE 2BY SBI D6 2BY SUI EE 2BY XRI
 9 : 3BY C;: C, , ;       CD 3BY CALL
10 C3 3BY JMP 3A 3BY LDA 2A 3BY LHLD 22 3BY SHLD 32 3BY STA
11 0 BCON B 1 BCON C 2 BCON D 3 BCON E 4 BCON H 5 BCON L 6 BCON M
12 7 BCON A 8 BCON BC 9 BCON DE 0A BCON HL 0B BCON SP 0B BCON PSW
13 0E BCON #0 0F BCON =0 10 BCON NC 11 BCON CY 12 BCON PO
14 13 BCON PE 14 BCON >=0 15 BCON <0
15
```

BLOCK : 14

```
 0 : ARGERR "  ARGUMENT ERROR - " QUESTION ;
 1 : 1RG DUP 0< OVER 7 > + IF ARGERR THEN ;
 2 : 1YS C;: SWAP 1RG + C, ;
 3 88 1YS ADC 80 1YS ADD A0 1YS ANA B8 1YS CMP
 4 B0 1YS ORA 98 1YS SBB 90 1YS SUB A8 1YS XRA
 5 : 1Y1 C;: SWAP 1RG 8* + C, ;
 6 05 1Y1 DCR 04 1Y1 INR C7 1Y1 RST
 7 : MVI 1RG 8* 06 + C, C, ;
 8 : MOV 1RG 8* SWAP 1RG + 40 + C, ;
 9 : 2RG DUP 8 < OVER 0B > + IF ARGERR THEN 8 - ;
10 : 1YP C;: SWAP 2RG 16* + C, ;
11 09 1YP DAD 0B 1YP DCX 03 1YP INX 0A 1YP LDAX
12 C1 1YP POP C5 1YP PUSH 02 1YP STAX
13 : CRG DUP 0E < OVER 15 > + IF ARGERR THEN 0E - 8* + C, ;
14 : LXI 2RG 16* 1+ C, , ; : RETC C0 SWAP CRG ;
15 : 3YC C;: SWAP CRG , ; C4 3YC CALLC C2 3YC JMPC
```

BLOCK : 15

```
 0 ( ABSOLUTE ASSEMBLER 4 OF 4 )
 1 : BEGIN PC @ ;
 2 : END 1 XOR JMPC ;
 3 : LAD BEGIN OFFSET + 2 - ;
 4 : IF 0 SWAP END LAD ;
 5 : THEN BEGIN SWAP ! ;
 6 : ELSE 0 JMP THEN LAD ;
 7 : WHILE BEGIN ;
 8 : PERFORM IF ;
 9 : PEND SWAP JMP THEN ;       0 CONSTANT AMODE
10 : SUBRTN FORTH <BUILDS PC @ , DOES> @
11   AMODE IF ASSEMBLER CALL FORTH ELSE ABS-ASMB CALL FORTH THEN ;
12 BASE C!
13 FORTH DEFINITIONS 16 2 LOADS  ( TBUG DUMP )
14 ( ENTRY-POINT TDUMP <NAME> )
15 : TDUMP PC @ ORG @ - ORG @ BAREA TAPE-DUMP ;
```

Then, KOUT CALL to call that routine.
Lines 10-11 allow set-up of a table of bytes of information (not used in this example).
Lines 12-13 initialize the Absolute Assembler by setting the variables PC and OFFSET and clearing BAREA. If you have changed the length of BAREA, change the length here, too.
Line 14 defines %, equivalent to $L but without its length byte. Example: LABEL STR % ABC% creates 3 bytes with contents ABC ; can be addressed by specifying STR .
Line 15 saves current BASE on stack and goes to HEX .

Second and Third Blocks (13-14): Define the Absolute Assembler 8080 instructions.

Fourth Block (15):
Lines 1-2 define BEGIN...END construct.
Line 3 defines a word to get last address used.
Lines 4-6 define IF...ELSE...THEN constuct.
Lines 7-9 define PERFORM...WHILE...PEND construct.
Lines 10-11 use <BUILDS and DOES> to define SUBRTN, which will cause a name to be applied to a subroutine. The subroutine then can be called simply by specifying the name. (Whereas a routine defined with the word LABEL must be called by specifying the name followed by CALL .)
Line 12 resets BASE as before.
Lines 13-15 provide Compile routines for creation of SYSTEM tapes.

Fifth and Sixth Blocks (16-17): This TBUG Dump routine creates SYSTEM tapes.

PLUS, AN EXAMPLE WHICH TESTS YOUR TAPE SYSTEM:

The example used here is a program which reads in tapes written by the WTST (write-test) Program (see Block 21) with a simple 12345..., etc. The purpose of the program is to ascertain if a TRS-80 cassette recorder is writing and reading properly,and over what range of volumes. Blocks 21 and 22 are MMSFORTH programs to write the tape and read it in again.
Block 20 is the Absolute Assembler version of Block 22. It allows one to run the test without MMSFORTH. The test data tape must already have been created by WTST, from Block 21 (using MMSFORTH).
Line 1: ABS-ASMB , same as 12 4 LOADS . But first, you must add this definition to the directory commands on Block 10:
: ABS-ASMB 12 4 LOADS ;
Then, this second use of ABS-ASMB invokes the Absolute Assembler vocabulary which is redefined in Block 12, Line 4. This way, if you already have loaded the Absolute Assembler it merely invokes the vocabulary twice instead of loading an additional copy.
6000 ORG ! sets 6000 hex (24576 decimal) as the location where the program will be loaded later.
INITASMB initializes the Absolute Assembler.
Line 2 defines the subroutine to ECHO a character to the screen by calling location 33 in the ROM.
Line 3 defines CR (carriage return) as a 0D hex (13 decimal) ECHO .
Line 4 defines CLS (clear screen) as 1C (home cursor) and 1F (clear to end-of-frame).
Line 5 defines routine to turn on the cassette recorder (same as Line 2 of Block 22).
Line 6 defines routine to turn off cassette recorder (same as Line 3 of Block 22).
Line 7 defines routine to synchronize on data (same as Line 4 of Block 22).
Line 8 defines routine to read data from tape (same as Line 5 of Block 22).
Line 9: LABEL RTST defines RTST (read-test) as a labeled routine. This is the program itself, from CLS to JMP on Line 12.
Line 13: RTST . prints the entry-point of the program. This is an optional routine and serves as documentation only.
RTST TDUMP RTST puts the name RTST on the SYSTEM tape and also dumps the compiled program from the work area (from ORG through current value of PC). Its entry point will be the beginning of the RTST routine in the program. The format of this sequence is:
(entrypoint) TDUMP (name)
Line 14: Clean up after the program, reset BASE , FORGET the program and return to Directory.

BLOCK : 16

```
 0 ( ENTRY-POINT  CNT   START-ADR  BUF-ADR  TAPE-DUMP <NAME> )
 1 BASE C@   HEX     0 CONSTANT OFST
 2 CODE TON     A XRA    212 CALL    NEXT
 3 CODE TOFF    1F8 CALL    NEXT
 4 CODE TWRITE-LEADER   BC PUSH   287 CALL   BC POP   NEXT
 5 CODE TCHAR-OUT   HL POP   L A MOV   264 CALL   NEXT
 6 CODE TWORD-OUT HL POP  L A MOV 264 CALL   H A MOV 264 CALL  NEXT
 7 CODE HI-BYTE  HL POP  H L MOV  0 H MVI  PSH
 8 ( ADDRESS CNT DATA-BLOCK )
 9 : DATA-BLOCK   3C TCHAR-OUT   DUP TCHAR-OUT ( CNT )
10   OVER TWORD-OUT ( ADR )  OVER DUP FF AND SWAP HI-BYTE +
11   ( CKSUM OF ADR )   ROT   OFST +   ROT
12   0   DO   DUP C@ ROT + SWAP    DUP C@ TCHAR-OUT   1+
13      LOOP   DROP   TCHAR-OUT ( CKSUM ) ;
14 ( ENTRY-POINT  ENTRY-OUT  )
15 : ENTRY-OUT 78 TCHAR-OUT TWORD-OUT ;
```

BLOCK : 17

```
 0 ( TAPE-DUMP ROUTINES 2 OF 2 )
 1 ( TAPE FORMAT: 255 ZEROES A5H 55H 6-BYTE FILENAME      )
 2 (              DATA-BLOCKS  78H  ENTRY-ADDRESS         )
 3 ( DATA-BLOCK FORMAT: 3CH  CNT  DATA  CHECKSUM           )
 4 (     CHECKSUM = 1 BYTE SUM OF ADDRESS AND DATA        )
 5 ( ENTRY-POINT  CNT   START-ADR  BUF-ADR  TAPE-DUMP <NAME> )
 6 : TAPE-DUMP  OVER - (') OFST !  TON  TWRITE-LEADER  55 TCHAR-OUT
 7   BL WORD HERE 1+ DUP HERE C@ 6 MIN + SWAP
 8   DO I C@ TCHAR-OUT LOOP
 9   HERE C@ 6 - 0< IF 6 HERE C@ - 0 DO 20 TCHAR-OUT LOOP THEN
10   SWAP  WHILE   100 - DUP 0 >=
11         PERFORM   OVER 100 DATA-BLOCK  SWAP 100 + SWAP
12         PEND
13   DUP -100 <> IF   100 + DATA-BLOCK  ELSE   DROP DROP   THEN
14   ENTRY-OUT   TOFF
15 ;     BASE C!
```

BLOCK : 20

```
 0 ( TAPE CALIBRATION ROUTINE RTST, USING ABSOLUTE ASSEMBLER )
 1 ABS-ASMB  ABS-ASMB  HEX  6000 ORG ! INITASMB
 2 SUBRTN ECHO BC PUSH 33 CALL BC POP RET
 3 SUBRTN CR 0D A MVI ECHO RET
 4 SUBRTN CLS 1C A MVI ECHO 1F A MVI ECHO RET
 5 SUBRTN TRON BC PUSH A XRA 212 CALL BC POP RET
 6 SUBRTN TROFF BC PUSH  1F8 CALL BC POP RET
 7 SUBRTN RSYNC BC PUSH  296 CALL BC POP RET
 8 SUBRTN TREAD BC PUSH  235 CALL BC POP RET
 9 LABEL RTST   CLS TRON
10   BEGIN  CR RSYNC 3C B MVI
11      BEGIN TREAD ECHO B DCR =0 END
12   JMP
13 RTST .  RTST TDUMP RTST
14 DECIMAL   FORGET TASK   DIR
15
```

BLOCK : 21

```
 0 ( WTST IN MMSFORTH )     : TASK ;
 1
 2 HEX  CODE TRON BC PUSH A XRA 212 CALL BC POP NEXT
 3      CODE TROFF BC PUSH  1F8 CALL BC POP NEXT
 4      CODE RSYNC BC PUSH  296 CALL BC POP NEXT
 5      CODE TREAD BC PUSH  235 CALL BC POP A L MOV 0 H MVI PSH
 6      CODE TWRITE HL POP L A MOV BC PUSH 264 CALL BC POP NEXT
 7
 8 : WTST TRON BEGIN 0 TWRITE 0 TWRITE A5 TWRITE
 9   6 0 DO 0A 0 DO I 30 + TWRITE LOOP LOOP " *" ?KEY END TROFF ;
10
11
12 DECIMAL
13 WTST  FORGET TASK  DIR
14
15
```

BLOCK : 22

```
 0 ( RTST IN MMSFORTH )         : TASK ;
 1
 2 HEX  CODE TRON BC PUSH A XRA 212 CALL BC POP NEXT
 3      CODE TROFF BC PUSH  1F8 CALL BC POP NEXT
 4      CODE RSYNC BC PUSH  296 CALL BC POP NEXT
 5      CODE TREAD BC PUSH  235 CALL BC POP A L MOV 0 H MVI PSH
 6      CODE TWRITE HL POP L A MOV BC PUSH 264 CALL BC POP NEXT
 7
 8 : RTST CLS TRON BEGIN CR RSYNC 3C 0 DO TREAD ECHO LOOP
 9    ?KEY END TROFF ;
10
11 DECIMAL    RTST  FORGET TASK  DIR
12
13
14
15
```

PICK & ROLL:

Some Forth users believe these two words are indispensable while others feel they are superfluous to the experienced Forth user. They are variable-depth stack-manipulation words. n PICK picks out a copy of the nth entry on stack and places it on top of stack, while n ROLL removes the nth entry from stack and places it on top of stack instead, rolling all higher entries down one in the process. So 2 PICK is equivalent to OVER, 2 ROLL is SWAP, 3 ROLL is ROT, etc. Got the idea? We have provided Forth versions for understandability and Forth 8080 Assembler versions for speed of execution. Try it, you may like it!:

BLOCK : 78

```
 0 ( PICK AND ROLL IN COMPILE FORTH - MMS/TD )
 1
 2 : PICK   2*  'S +  @ ;                              ( 18 BYTES )
 3
 4 : ROLL   DUP 1 >  IF 2*  DUP <R  'S +  DUP @ SWAP 1 -  DUP 2+ R>
 5          -MOVE SWAP DROP  ELSE DROP  THEN ;         ( 58 BYTES )
 6
 7
 8
 9
10
11
12
```

BLOCK : 79

```
 0 ( PICK AND ROLL IN ASSEMBLER FORTH - MMS/TD )
 1
 2 CODE PICK  HL POP  HL DCX  HL DAD  SP DAD  M E MOV  HL INX
 3      M D MOV   DE PUSH  NEXT                        ( 19 BYTES )
 4
 5 CODE ROLL  HL POP  L A MOV  HL DCX  HL DAD  SP DAD  HL INX
 6      L E MOV  H D MOV   BC PUSH
 7      M B MOV  HL DCX  M C MOV  HL DCX  BC PUSH  A C MOV
 8        WHILE   C DCR  #0
 9        PERFORM M A MOV  DE STAX  HL DCX  DE DCX
10                M A MOV  DE STAX  HL DCX  DE DCX
11        PEND
12      XCHG  BC POP  B M MOV  HL DCX  C M MOV  BC POP
13 NEXT                                              ( 48 BYTES )
14
15
```

MMSFORTH QUICKIES

This month's program quickies consist of short utility routines suitable for dashing off from keyboard or temporarily patching into programs during development. Some are contributed, others provided by MMS as the result of user suggestion or our own programming usage. The usual response from first-time users has been, "Great! You ought to incorporate this into your next version of MMSFORTH!" In generaI, we disagree. As you get to know Forth you too will dash off the right one for the job when it is appropriate, and not clog up the system when it is not. Besides, next month you're going to send us improved versions and a new rash of "necessaries", right?

PRINT, UNSIGNED:

This quickie eliminates the beginner's confusion or expert's impatience with negative values returned by Forth for single-precision numbers greater than 32K. It also applies MMSFORTH's pictured numeric output in an interesting manner, and may inspire you to experiment further with these PRINT USING-like words.

```
: .U   SPACE  <# #S #>  TYPE ;
```

PRINT STACK:

Plant this beauty in your development package to report the immediate situation in tricky places, or call it from the keyboard during interpretive debugging. It gives you a stack printout, but leaves the stack itself as is. Note that we first find the stack's base address, S0, and that we don't subtract 'S from S0 . Why not? Because in Forth, S0 'S - will first add S0 to the stack, yielding a larger answer for 'S .

```
: S0   19203 @  256 - ;
: .S   'S S0 -  IF 'S S0 SWAP DO  I ?  2 +LOOP  THEN ;
```

Forth beginners are encouraged to use .S to familiarize themselves with Forth's stack manipulator words:

```
1 2 3 .S yields 3 2 1
SWAP .S         2 3 1
ROT .S          1 2 3
OVER .S         2 1 2 3
DROP .S         1 2 3
- .S           -1 3 ..., etc.
```

Want those unsigned numbers, instead? Just load the .U definition first, then substitute I @ .U for the I ? in .S . All the necessary clues for finding S0 are in the Memory Map on Page 3 of Newsletter #1. We are indebted to Jim Gerow for a much larger routine which did this and also printed the ASCII character equivalent, if any. For sheer simplicity and minimum code, Dick Landau offers a rough-cut version suitable for some uses:

```
: .S   'S 10 DUMP ;    Now, THAT's compact!
```

AVAILABLE MEMORY:

We mentioned this one in Newsletter #1. It fits in nicely here, using our new .U in place of . :

```
: MEM   'S PAD - .U ;
```

ASCII:

Tom Dowling provided this definition after a suggestion by Paul van der Eijk of the Washington, D.C.-area user group. Follow ASCII by any ASCII-character to put its code on top-of-stack. This "intelligent" definition may be used in compiled or immediate-execute modes, and can improve the readability of some routines.

```
: ASCII   IMMEDIATE  BL WORD  HERE 1+ C@
          STATE C@  IF (L) (L) , ,  THEN ;
Example:  ASCII A .  yields 65 .
```

MMSFORTH MODIFICATIONS

MMSFORTH SYSTEM:

While creating the Forth version of this month's KALEIDOSCOPE program, I discovered an old mistake lurking deep within Block 32 of the MMSFORTH System. This block is accidently concluded with a ;S (required in some versions of Forth but optional in ours). We're modifying our statement in the MMSFORTH Glossary to expIain that ;S concludes the loading not only of the present block but of any other blocks within a single LOADS command. That's why our KALEIDOSCOPE block loads Blocks 32 and 33 separately instead of 32 2 LOADS . After you remove the ;S from the end of your system's Block 32, you can use the 32 2 LOADS command successfully.

THE DATAHANDLER:

Here are two improvements to add to copies of THE DATAHANDLER V1.1 produced prior to 8/19/80 (see date on original write-protect tab). These changes eliminate a rare junk read-out following a SORT on a numeric field.

First, several routines now use temporary scratch space 512 bytes above PAD , so be sure to leave adequate RAM space to accommodate them without interfering with the stacks. Load THE DATAHANDLER, press Break, and enter: 'S PAD - . to display the remaining available bytes of RAM. MMS recommends leaving at least 650 bytes; if this test reads less, either delete some source programming or set M#BLOCKS (at extreme left of Block 40, Line 2) to a lower number of 1K-byte blocks.

Secondly, the DATAHANDLER routine, VAL (Block 25, Lines 0 and 1), should be moved up 512 bytes to that area, as follows:

```
 0 : VAL  FIELD PAD 512 + $!  0  PAD 512 + DUP C@ +  1+ C!  PAD 512
+
 1   NUMBER DROP #PT C@ 0= IF HI# @ SWAP 1 #PT C! THEN  ;
```

(Some custom versions define 2VAL instead, and will require equivalent changes thereto.)

After changing either or both of the above items reload the source blocks and, if appropriate, reprecompile as well.

THE LAST WORD: "Go Forth and COMPILE!"
- from the newest testament (USING FORTH?)

MMS MILLER MICROCOMPUTER SERVICES
61 LAKE SHORE ROAD, NATICK, MASS. 01760
(617) 653-6136

==========
FIRST CLASS MAIL
U.S. POSTAGE
PAID
Framingham, MA 01701
Permit No. 207
==========